VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Návrh a vytvoření webové aplikace pro správu investic

Design and Creation of a Web Application for Investment Management

Student: Jan Kožušník

Vedoucí bakalářské práce: Ing. Martin Pochyla, Ph.D.

Ostrava 2018

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra aplikované informatiky

Zadání bakalářské práce

Student: **Jan Kožušník**

Studijní program: B6209 Systémové inženýrství a informatika

Studijní obor: 6209R017 Informatika v ekonomice

Téma: Návrh a vytvoření webové aplikace pro správu investic
Design and Creation of a Web Application for Investment Management

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Úvod
2. Teoretická východiska realizace webové aplikace
3. Analýza současného stavu a požadavků
4. Návrh a vytvoření webové aplikace
5. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků bakalářské práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

STROUKAL, Dominik a Jan SKALICKÝ. *Bitcoin: peníze budoucnosti: historie a ekonomie kryptoměn, stručná příručka pro úplné začátečníky.* Praha: Ludwig von Mises Institut CZ&SK, 2015. ISBN 978-80-87733-26-4.
HOPKINS, Callum. *PHP okamžitě.* Brno: Computer Press, 2014. ISBN 978-80-251-4196-0.
SHARKIE, Craig a Andrew FISHER. *Responzivní webdesign: okamžitě.* Brno: Computer Press, 2015. ISBN 978-80-251-4384-1.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Pochyla, Ph.D.**

Datum zadání: 24.11.2017
Datum odevzdání: 11.05.2018

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * OSTRAVA * Ekonomická fakulta 1

Ing. Petr Rozehnal, Ph.D.
vedoucí katedry

prof. Dr. Ing. Zdeněk Zmeškal
děkan fakulty

Prohlašuji, že jsem celou práci, včetně všech příloh, vypracoval(a) samostatně.

V Ostravě dne 7.5.2018

.....

Jan Kožušník

Rád bych poděkoval panu Ing. Martinu Pochylovi, Ph.D. za odborné vedení mé bakalářské práce, cenné rady, připomínky a čas, který mi věnoval.

Obsah

1 Úvod

V dnešním silně konkurenčním prostředí je nezbytné, aby se firmy snažily držet krok s dobou, nezaostávaly za konkurencí, nabízely svým klientům minimálně stejně kvalitní nebo kvalitnější služby, než konkurence a měly dostatek času na rozvoj svého podnikání a posouvání se kupředu.

Pokud chceme firmu rozvíjet a posouvat dál, rozhodně toho nedosáhneme trávením spousty času nad administrativou a podobnými činnostmi, které nám mohou dnešní moderní technologie usnadnit a zjednodušit. Za tímto účelem v posledních letech vzniká spousta online systémů – například CRM systémy, fakturační systémy nebo DMS, které mají za cíl usnadnění podnikání, zjednodušení přístupu k informacím a zjednodušení různých činností, ať už vedlejších nebo nezbytných pro chod podniku.

Díky pokroku v oblasti chytrých zařízení jsou navíc tyto systémy z velké části dostupné také z mobilního telefonu či tabletu, a je jedno, zda je uživatel právě v kanceláři nebo na pláži.

Cílem mé bakalářské práce je vytvoření online systému pro investiční společnost za účelem zjednodušení a zpřehlednění správy zákazníků, obchodních zástupců a investic a zjednodušení, potažmo automatizování, určitých procesů ve společnosti. Do tohoto systému budou mít přístup jak klienti, tak také obchodní zástupci a vedení společnosti, přičemž dle typu uživatele se bude částečně měnit také uživatelské rozhraní systému.

Při vývoji systému chci využít moderních technologií, a to z několika hlavních důvodů. Prvním z nich je zjednodušení samotného vývoje. Není potřeba se zabývat úplnými základy vývoje systému použitím PHP frameworku, ať už se bavíme o autentizaci a autorizaci uživatelů, přístupu k databázi, routování. Dále například mnou používaný framework obsahuje také nástroj pro debugování a logování, díky čemuž jsem schopen mnohem lépe odhalit případné problémy, co systém brzdí. Práci si lze zjednodušit i na frontendu, například jQuery knihovnou, díky níž je práce s JavaScriptem o dost jednodušší. Samozřejmě je kvůli těmto "nadstavbám" provádění jednotlivých příkazů pro zařízení (ať už na straně uživatele nebo na straně serveru) náročnější, ale na druhou stranu, současná zařízení jsou natolik výkonná, že případné užívání těchto prostředků v rozumné míře není na škodu. Druhý důvod souvisí s jednoduchostí užívání

pro uživatele - jednak díky využití možností, jež práci zjednoduší, je více času na samotnou optimalizaci systému tak, aby byla práce s ním pro uživatele jednodušší, ale hlavně tyto technologie samy o sobě poskytují způsoby, jak zjednodušit uživateli používání systému. Jedná se kupříkladu o validaci formulářů před samotným odesláním na server, dynamicky generované tabulky, grafy či kalkulačky, které jsou schopny počítat hodnoty automaticky hned při zadávání.

2 Teoretická východiska realizace webové aplikace

V této kapitole nastíním aktuální trendy ve vývoji webových aplikací, rozeberu výhody a nevýhody webových aplikací a rozepíši základní technologie a prostředky, které budou použity při vývoji webové aplikace.

2.1 Aktuální trendy ve vývoji webových aplikací

Pro aktuální trendy ve vývoji webových aplikací je charakteristické:

- Responzivita a použitelnost na širokém spektru zařízení – od mobilních telefonů, přes tablety až po PC. Optimalizace pro pohodlné použití na různorodých zařízeních je dnes již prakticky nedílnou součástí tvorby většiny webových aplikací.
- Minifikace – Proces minifikace spočívá ve zmenšování velikosti výsledných souborů – u JavaScriptových souborů a CSS souborů dochází zpravidla k odstranění bílých znaků, takže tyto soubory potom vypadají jako shluk znaků bez větších mezer, napsaných většinou na jednom řádku. Díky tomu se sníží jejich velikost a jejich načtení je rychlejší.
- AJAX, Lazyloading – Lazyloading je technika, která urychluje načítání samotného webu, a to tak, že se obsah načítá až tehdy, kdy je potřeba. Jako příklad užití poslouží sociální síť Facebook. Kdyby měla webová aplikace Facebook načítat všechny komentáře u příspěvku, načtení by trvalo zbytečně dlouho. Proto se komentáře zpravidla načítají až po kliknutí na nápis „Komentáře“ u konkrétního příspěvku. Tím se zrychlí načtení hlavního obsahu a také sníží počet načítaných dat a odlehčí serveru, protože je nepravděpodobné, aby si návštěvník u každého příspěvku zobrazoval komentáře.

  Techniku lazyloadingu implementují frameworky Angular či React.js, které nabízí již předpřipravené mechanismy a funkce pro vytvoření takové webové aplikace. Na technologii AJAX se zaměřím níže.

S moderním vývojem webových aplikací souvisí pojem „Progressive web apps“. Pod tímto označením můžeme hledat webové aplikace, jež se pro své uživatele tváří, jako by byly mobilními aplikacemi, které si na své zařízení nainstaluje.

Pro progresivní webové aplikace je charakteristické, že by měly být rychlé – bez pomalých odezev ze strany serveru a jsou často doplněny o animace, díky kterým aplikace působí plynulejším dojmem. Dále je pro tyto aplikace typické, že je lze spustit z domovské obrazovky bez nutnosti instalace – na domovskou stránku si uživatel uloží „odkaz“ na webovou stránku s danou aplikací. Tato aplikace může mít nastaveny parametry (buď pomocí HTML atributů v hlavičce nebo pomocí speciálního souboru), díky nimž může vývojář ovlivnit, jak se samotná aplikace bude uživateli zobrazovat – například, byť se aplikace otevírá v prohlížeči, může u ní vývojář schovat standardní navigační menu.

Posledním charakteristickým bodem je možnost spuštění aplikace také v offline režimu. Aplikace musí umět pracovat s JavaScriptem a cache. Díky tomu lze uživateli poskytnout obsah, i když si stránku, respektive webovou aplikaci, načte pouze jednou, a poté ji používá offline.

2.2 Výhody a nevýhody online systému

Mezi hlavní výhody řadím fakt, že samotnou webovou aplikaci nemusíme instalovat – stačí otevřít webový prohlížeč, zadat adresu aplikace a můžeme ji začít používat (neuvažujme aplikace, požadující Silverlight, Flash a podobné technologie). Další nespornou výhodou je možnost spustit tuto aplikaci odkudkoliv – ať už se budeme bavit o místě - doma, v práci či v kavárně nebo o zařízení – mobilní telefon, tablet, PC, na kterých navíc nejsme omezeni operačními systémy. Samozřejmě jsou výjimky, jako aplikace, které spustíte pouze v rámci firemní sítě nebo aplikace, které jsou optimalizovány pouze pro použití na PC a ještě k tomu v prohlížeči IE 8, nicméně obecně se vývojáři snaží o maximální univerzálnost použití.

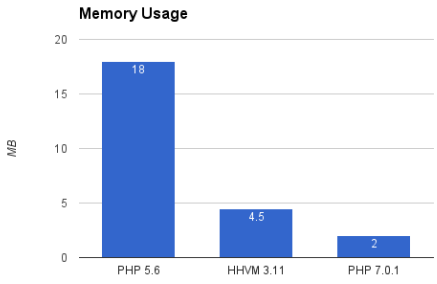
Naopak jako zásadní nevýhodu webové aplikace vnímám fakt, že prakticky pokaždé, chcete-li tuto aplikaci využívat, musíte být připojeni k internetu. Další nevýhodou je výkon – aplikace, která bude napsána například v C# a bude spuštěna na desktopu bude rychlejší než aplikace, která bude potřebovat Frontend, na němž poběží například JavaScript, a poté Backend, který bude ať už v PHP, Java či nějakém jiném jazyce – už jenom z toho důvodu, že si budete muset vyhradit čas na komunikaci mezi zařízením uživatele a serverem. Samozřejmě tento fakt lze obejít dobrou prací

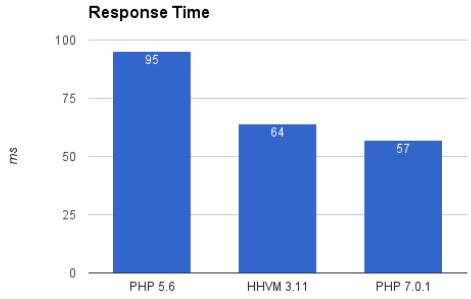
s JavaScriptem, přičemž o odesílání dat na server nemusí uživatel vůbec vědět, ale stále je tam minimálně prostředník, tj. prohlížeč, který si také bere kus výkonu.

2.3 PHP a frameworky

PHP, celým názvem „Hypertext Preprocessor“, je programovací jazyk, jež se hojně využívá pro psaní webových aplikací. Historie tohoto jazyka sahá do poloviny 90. let minulého století. V současné době je aktuální verze jazyka PHP 7.2. Obecně verze 7.x přináší do jazyka snížení paměťové náročnosti, ale primárně velké zrychlení samotných aplikací napsaných v tomto jazyce.

Porovnání verzí jazyka PHP popisují následující obrázky 1 a 2 . Jedná se o verzi PHP 5.6 s povoleným opcache, PHP s HHVM 3.11, a poté PHP 7.0.1.. Jedná se o porovnání paměťové náročnosti a potřebného času pro načtení stránky s „top příběhy“ v CMS systému eZ Platform, postaveném na frameworku Symfony 2.7.8.





Obrázek 1 Využití paměti (zdroj: Jani Tarvainen,2015) *Obrázek 2 Doba odpovědi (zdroj: Jani Tarvainen,2015)*

Facebook pro zrychlení PHP vyvinul vlastní řešení, které se nazývá HHVM. HHVM je virtuální stroj, který funguje na principu popsaném v citaci:

„Prvním krokem je kompilace zdrojového kódu PHP (nebo také jazyka Hack, který je od PHP odvozen) do bytecode (HHBC). Ten se pak před spuštěním překládá do strojového kódu a optimalizuje. Výsledkem je rychlejší běh jak oproti přímé interpretaci, tak oproti interpretaci bytecode. Na druhou stranu je potřeba podstatně více

paměti, s čímž je nutno počítat.“
(Lukáš Jelínek, 2015)

Při vývoji webových aplikací je populární používat frameworky. Jsou to předpřipravené kusy kódu, které řeší spousty věcí za samotného programátora. Rozsah těchto funkcí záleží na frameworku, nicméně zpravidla frameworky už mají implementovanou práci s databází, sessions a cookies, routování a základní práci s autentifikací a autorizací. Frameworky mívají již často implementovány bezpečnostní mechanizmy jako hashovací funkce pro práci s hesly nebo ochranu proti CSRF útokům.

Další významnou výhodou je implementace cachování, ať už výstupů z šablon nebo výsledků z databáze.

Nevýhodou frameworků je fakt, že mohou být pro potřeby programátora příliš velké a zbytečně se tak načítá spousta souborů, což běh této aplikace brzdí. Tuto nevýhodu frameworky částečně zmírňují propracovanými mechanismy pro cachování a také možností, že si lze u některých frameworků nastavit, které soubory se mají automaticky načítat a které nikoliv.

2.4 CSS a LESS

CSS, Cascading Style Sheets nebo také „Kaskádové styly“, je jazyk, jež se používá pro stylování obsahu webových stránek. Samotné stylování funguje na principu tzv. selektorů a následného zápisu stylů, jež se mají aplikovat. Tyto styly se zapisují do složených závorek.

```
.cerveny {color: red;}
.velky-text {font-size: 30px;}
#fialovy {color:purple;}
input[type=”text”] {border-radius: 3px; outline: none;}
.clanek p:nth-child(2n+1) {float: left; width: 50%;}
.clanek p:nth-child(2n) {float: right; width: 50%;}
```

Zdrojový kód 1 Ukázka CSS selektorů

V kódu Zdrojový kód 1 jsou patrné různé druhy CSS selektorů. Selektory s tečkou na začátku: `.cerveny`, `.velky-text` jsou selektory, kterými vybíráme prvky na základě atributu class. Viz například tato ukázka použití stylů:

`<p class="cerveny velky-text">Červený text, velikosti 30px</p>`. Selektory, začínající mřížkou - `#fialovy` – jsou selektory podle atributu id – například `<div id="fialovy">Tento text bude fialový</div>`. Dále vidíme ukázku selektoru podle atributu HTML tagů – podle tagu `type`. Zbývající dva selektory používají tzv. pseudoselektory, tj. selektory, které nejsou přímo zakomponovány v HTML kódu, ale CSS si je umí odvodit.

V současnosti je populární používat pro psaní CSS stylů jazyky LESS a případně SASS. Jedná se o jazyky, vycházející z CSS, nicméně umožňující vývojáři snadnější zápis stylů. LESS a SASS jsou jazyky, které slouží pouze pro zápis stylů, nelze je ale přímo aplikovat pro stylování samotné stránky. Pro použití je třeba je převést na CSS. To lze provést automatizovaně například v IDE nebo pomocí GRUNTu, skrze online kompilátor nebo pomocí JavaScriptu. Poslední řešení je ovšem vhodné výhradně pro vývojářské účely, protože pro aplikování stylů je potřeba načíst stránku a JavaScriptový kód, překompilovat kód do CSS, a až poté se aplikují styly.

Možnosti, jež LESS či SASS nabízí se trochu liší, nicméně v zásadě lze říci, že můžeme používat proměnné, funkce, předpřipravené styly, tzv. mixins či výpočty. Významnou výhodou je možnost zanořování stylů do sebe.

```
/* zapis v CSS */
.clanek {
	background: #f9f9f9;
}
.clanek .nadpis {
	font-weight: bold;
}
.clanek .popis {
	font-style: italic;
}
.clanek p:first-of-type {
	font-size: 1.1em;
}
/* zapis v LESS */
.clanek {
	background: #f9f9f9;
	.nadpis {
		font-weight: bold;
	}
	.popis {
		font-style: italic;
	}
	p {
		&:first-of-type {
			font-size: 1.1em;
		}
	}
}
```

Zdrojový kód 2 Porovnání CSS a LESS

Výše uvedená ukázka ukazuje, jaký je rozdíl v zápisu stylů pomocí CSS a LESS.

2.5 JavaScript, jQuery

JavaScript je programovací jazyk, jež se používá pro programování na straně návštěvníka webu. Díky tomuto jazyku můžeme web udělat dynamickým, aniž bychom museli návštěvníka odkazovat na jinou stránku. Za dynamický se dá web považovat ze dvou odlišných pohledů:

- Při požadavku uživatele načte data ze serveru, aniž bychom museli uživatele přesměrovat.

- Při požadavku uživatele – například kliknutí na tlačítko – můžeme změnit obsah stránky, zobrazit uživateli vyskakovací okno či validovat vyplněné prvky formuláře.

Díky výše zmíněným způsobům užití se stává tento jazyk a jeho aplikace v poslední době velmi populárními. Tento fakt koresponduje s výše zmíněnými trendy ve vývoji webových aplikací, kde se snažíme o co nejpříjemnější prostředí pro uživatele, čehož z velké části dosahujeme právě pomocí tohoto jazyka.

V současné době se jazyk JavaScript dokonce začíná používat na straně serveru, viz kupříkladu Node.JS.

Z důvodu popularity JavaScriptu se rozhodli vývojáři z Google vytvořit knihovnu, jež se jmenuje jQuery. Díky této knihovně může vývojář psát v JavaScriptu, nicméně mnohem jednodušším způsobem, protože jQuery obsahuje již předdefinované funkce.

Asi nejvýznamnější výhodou jQuery oproti JavaScriptu je použití selektorů. Tyto selektory vychází ze selektorů používaných v CSS.

```
/* verze v "cistem" JavaScriptu */
var elements = document.getElementsByClassName("trida");
var i;
for(i = 0; i < elements.length; i++) {
   elements[i].style.backgroundColor = "red";
}

/* verze v jQuery" */
$(".trida").css({"background-color":"red"});
```

Zdrojový kód 3 Porovnání JavaScriptu a jQuery

Na výše uvedeném kódu vidíme ukázku, jak pomocí „čistého“ JavaScriptu, někdy také nazývaného jako „vanilla JS“ a pomocí jQuery změnit barvu pozadí pro elementy s třídou „trida“.

Je zřejmé, že použití jQuery je oproti JavaScriptu mnohem jednodušší. Velkou nevýhodou jQuery je fakt, že je to knihovna, která funkce, jež v ní voláte, nahrazuje funkcemi v JavaScriptu, což dělá program napsaný v jQuery pomalejším než program čistě v JavaScriptu.

2.6 MariaDB

Pro správu dat bude použita relační databáze MariaDB, která vychází z databáze MySQL. Databáze MariaDB začala vznikat v období koupě MySQL firmou Oracle z důvodů obav o budoucí vývoj této databáze.

2.7 Composer

Při vývoji webových aplikací v jazyce PHP, obzvláště pokud aplikaci stavíme na frameworku, se velmi často dostaneme do styku s takzvaným composerem.

Composer je nástroj pro správu „balíčků“ v jazyce PHP, které můžeme do našeho projektu vkládat. Například pokud chceme ve webové aplikaci pracovat s PDF dokumenty, respektive je generovat, můžeme si napsat vlastní knihovnu pro práci s PDF, ručně stáhnout již připravenou knihovnu anebo tuto knihovnu napsat do konfiguračního souboru pro composer, a poté spustit přes příkazový řádek příkaz, díky kterému se tato knihovna automaticky vloží do frameworku a my ji můžeme ihned používat.

Pro composer je charakteristický konfigurační soubor, ve kterém si nastavíme parametry samotného projektu (název, popis, licenci apod.) nebo také knihovny, jež do projektu chceme vložit. Tento soubor je v JSON formátu a jeho strukturu popisuje následující zdrojový kód.

```
{
    "name": "Název projektu",
    "description": "Popis projektu.",
    "keywords": ["framework", "laravel"],
    "license": "MIT",
    "type": "project",
    "require": {
        "php": ">=5.5.9",
        "laravel/framework": "5.2.*",
        "barryvdh/laravel-debugbar": "~2.4",
        "laravelcollective/html": "5.2.6"
    }
}
```

Zdrojový kód 4 Konfigurační soubor pro composer

V tomto kódu jsme nastavili název projektu, jeho popisek, klíčová slova, licenci, typ a poté požadavky, jež je třeba splnit, abychom mohli projektu spustit:

- PHP verze vyšší nebo rovné 5.5.9.
- Laravel framework verze 5.2.*, respektive libovolné verze 5.2.
- Knihovnu Laravel debugbar ve verzích minimálně 2.4 až verzi < 3.0.
- Knihovnu pro práci s HTML formuláři apod. v konkrétní verzi 5.2.6.

2.8 AJAX

AJAX, neboli Asynchronous Javascript XML, je technologie, díky které můžeme posílat dotazy na server, aniž by byla potřeba samotnou stránku znovu načítat. Díky tomu jsme schopni vytvářet aplikace, jež jsou pro uživatele velmi příjemné na užívání a zároveň jsme schopni samotnou stránku zrychlit, a to hlavně díky již zmíněnému lazyloadingu, kde jednotlivé části stránky načítáme postupně, jakmile je potřebujeme.

V názvu je zmíněné slovo „asynchronous“, jež dobře vystihuje následující citace:

*„V čem vlastně spočívá **asynchronnost** v AJAXu? Jde o schopnost Javascriptu zavolat na serveru nějaký skript, nebo prvek API a nečekat bezprostředně na odpověď. Místo toho pokračuje provádění kódu (uživateli se například zobrazí stránka a může s ní běžně pracovat). Když potom odpověď přijde, provádění hlavního kontextu kódu se pozastaví (dříve, či později v závislosti na prioritě prováděné úlohy) a dojde k zavolání tzv. **callbacku** – funkce, která se provede v případě, že ze serveru dostaneme odpověď. Takovémuto kódu, či takovéto funkci se pak říká **neblokující**, protože neblokuje průběh provádění skriptu čekáním na odpověď, či zpracování.“*
(Taskkill)

AJAX podporuje samotný JavaScript, nicméně jeho použití je mnohem jednodušší pomocí jQuery, jež v sobě implementuje funkce `ajax`, `post` a `get`, kterými se pošle dotaz na server. Zároveň má vývojář připraveny callbacky `done` a `error`, které jsou po dokončení dotazu volány jQuery, podle výsledku odpovědi – pokud vše proběhne v pořádku, zavolá se callback `done`, v opačném případě je volán callback `error`.

2.9 API

API, neboli Application Programming Interface, je rozhraní pro programování aplikací. Ve své podstatně umožňuje vývojářům zajištění komunikace mezi dvěma na sobě nezávislými aplikacemi.

V současnosti je velmi populární tzv. REST API, což je de facto architektura, jak API budovat. Pro REST API je charakteristické, že právě akci, jež chceme provést, specifikujeme metodou dotazu – GET, POST či DELETE. Objekt, nad kterým chceme danou akci provést je specifikován na základě endpointu.

```
GET /projects           // vrátí seznam projektů
GET /projects/1234      // vrátí informace o určitém projektu
```

Zdrojový kód 5 Ukázka endpointů pro REST API (zdroj: Drahomír Hanák)

U API je důležité si dávat pozor, aby případní útočníci server, na který skrze API posíláme dotazy, nezahltili velkým množstvím dotazů. Proto je potřeba provádět ověřování, zda má program právo dotaz provést či nikoliv. K tomuto ověřování dochází z pravidla na základě API klíče, který se posílá zároveň s dotazem. Protože ale samotný API klíč není velký problém získat, některé webové stránky, poskytující API, implementují ještě omezení na množství dotazů, a to většinou počtem dotazů, jež můžeme s daným API klíčem provést za určitý časový úsek. Občas se setkáme také s propracovanějším řešením, kdy každý dotaz má bodové ohodnocení, které odráží jeho náročnost a s daným klíčem je asociován určitý počet bodů, které může za určitý časový úsek vyčerpat.

2.10 Kryptoměny

„Bitcoin je digitální P2P měna. Kryptoměna. Na rozdíl od současných peněz, jako jsou české koruny nebo americké dolary, nemá bitcoin žádnou centrální autoritu, která by se za něj zaručovala nebo měla možnost „tisknout“ nové peníze. Mimo této vlastnosti jde však o peníze se všemi standardními charakteristikami dobrých peněz. a mnohem více.“
(STROUKAL, 2015, s.19)

Byť se citace zmiňuje pouze o Bitcoinu, vlastnosti, jež popsala, lze z velké části aplikovat také na ostatní kryptoměny, kterých je nyní více než 1 500. Mezi ty kvalitnější

můžeme zařadit sice pouze zlomek z nich, nicméně toto číslo odráží velkou popularitu kryptoměn.

Kryptoměny vděčí za svou popularitu velké rychlosti provádění transakcí, nízkým poplatkům za transakce a také tomu, že je lze použít jako sice velmi riskantní investice, nicméně s obrovským zhodnocením. Pro příklad se můžeme podívat na loňský rok, kdy Bitcoin na začátku roku stál zhruba 1 000$ a ke konci roku atakoval hranici 20 000$. To jsou důvody, kvůli kterým se někteří neváhali zadlužit půjčkou či hypotékou na dům. Popularita kryptoměn neroste ovšem jen díky výše zmíněné obrovské výnosnosti, najde se také velká skupina investorů, jež kryptoměny obdivují díky technologiím, jež jsou s nimi spjaty – ať už se bavíme o blockchainu, smart kontraktech či lighting network.

Pro kryptoměny je charakteristický blockchain. Jde o databázi, ve které jsou zaznamenávány všechny transakce, které se uskutečnily. Tyto transakce jsou potom zpětně ověřovány pomocí speciálních programů, které běží na počítačích k tomu určených. Tyto počítače se nazývají ASICy a nebo RIGy. Proces ověřování se nazývá těžba, a to, zda se těží na ASICu nebo RIGu, závisí na algoritmu, na jakém daná měna funguje a také jaká je náročnost těžby. Například Bitcoin kvůli jeho vysoké náročnosti nelze na RIGu těžit.

ASIC je, jednoduše řečeno, počítač, který je konstruován tak, aby těžil měny pouze na jeden algoritmus. Díky tomu je velice efektivní, ale je bohužel svázaný pouze s jedním algoritmem, takže těžař nemůže reagovat na změny náročností těžby a přepínat mezi těženými měnami. RIGy jsou naopak počítače, disponující výkonnými grafickými kartami, na kterých se provádí výpočty a jejich provozovatel si může zvolit, který algoritmus použije. Těžař je tak schopen flexibilně reagovat na vývoj náročnosti těžby.

Transakce jsou prováděny tak, že se odešle z peněženky na určitou adresu zvolené množství například Bitcoinů. Tato informace se zašle do sítě, ve které „těžaři“ svým výpočetním výkonem ověřují to, že byly zaslány Bitcoiny, které jsou „jedinečné“ a že nemohou být například smyšlené. To probíhá na principu ověřování starých transakcí. Zjednodušeně se „těžaři“ snaží najít znění starých transakcí, kterým odpovídá hash, jež mají k dispozici. Tzn. dostanou hash, který vznikl na základě starých transakcí a oni zkouší tyto transakce zpětně podle hashe dohledat. Tyto transakce lze dohledat

pouze tak, že počítač zkouší různé kombinace transakcí tak dlouho, dokud jejich hash neodpovídá hashi, který mají k porovnání. Ten, který to jako první zvládne, pošle informaci o transakcích ostatním těžařům. Ti tuto informaci ověří a v případě její správnosti těžař, jež na tuto kombinaci přišel, dostane odměnu. Výše odměny se s postupem času liší. Byť se snaží těžit velké množství těžařů, odměnu získává pouze jeden. Z tohoto důvodu se těžaři sdružují do tzv. poolů, ve kterých sdílejí svůj výpočetní výkon a v síti vystupují jako jeden velice výkonný počítač, díky čemuž zvyšují svou šanci na uhodnutí správné kombinace transakcí. Při zisku odměny se odměna rozdělí mezi těžaře v poolu podle jejich podílu na celkovém výkonu poolu.

3 Analýza současného stavu a požadavků

3.1 Současný stav

V počátečním stavu vše fungovalo skrze administraci, postavené na WordPressu, kde se jednotliví obchodní zástupci registrovali a měli tak přístup do sekce pro přihlášené uživatele. Zde mohli vidět tabulku s výpisem investic a grafické zobrazení výnosů jednotlivých investičních produktů. Data do této tabulky a grafu se vkládala skrze excelový soubor, který se nahrál na FTP.

Klient tedy musel pravidelně upravovat excelovou tabulku, do které připisoval výnosy/ztráty a nahrával ji na FTP. Dále musel pro každého obchodního zástupce, klienta a investici vytvářet všechny smlouvy manuálně. Každý den bylo nutné kontrolovat bankovní účet z důvodů vkladů do investic, či manuálně evidovat a zpracovávat požadavky na výběry a ukončení investic.

3.2 Požadavky

Základní požadavky ze strany klienta, týkající se funkčnosti, vyplývají z výše zmíněného a jsou to:

- Jednoduchá a přehledná evidence klientů, investic a obchodních zástupců.
- Možnost hromadného připisování výnosů a ztrát u investic.
- Automatizované zpracování požadavků na výběr.
- Automatizovaná kontrola vkladů na bankovní účet a následná aktivace investic či přípis vkladu.
- Automatizované generování smluv.

Na tyto základní požadavky postupně navazovalo například:

- Generování provizních sestav pro jednotlivé obchodní zástupce a implementace hierarchické struktury obchodních zástupců.
- Vytvoření „office“, do kterého měli přístup pouze obchodní zástupci a kde by měli nástěnku pro vkládání dokumentů nebo přidávání příspěvků.
- Vytvoření rezervačního systému pro rezervaci kancelářských prostor.

- Napojení na pool pro těžbu kryptoměn.
- Implementace správy školení pro nové obchodní zástupce, vč. propojení na „demo systém“ – obchodní zástupci, jež se přihlásili na školení, se automaticky propíší také do „demo systému“, kde si mohou zkoušet práci s klienty a investicemi, aniž by zasahovali do dat v reálném systému. V souvislosti s tímto bylo třeba také zajistit aktuálnost dat v „demo systému“.
- Jednoduchou možnost komunikovat se sekretářkou dle typu požadavku.
- Zobrazení výkonosti obchodních zástupců.

Tyto funkční požadavky byly doplněny o požadavky na vzhled webové aplikace, jež měla působit jednoduchým, avšak moderním dojmem a měla fungovat na mobilních zařízeních. Z technického hlediska měl klient pouze požadavek, aby aplikace fungovala skrze webový prohlížeč a aby ji mohl používat na webhostingu.

4 Návrh a vytvoření webové aplikace

4.1 Volba technologií

Volba použitých technologií v sobě odrážela s jakými technologiemi mám zkušenosti a také požadavky klienta.

Při volbě technologií byl hlavní důraz kladen na maximální univerzálnost použití – spuštění aplikace nezávisle na typu zařízení a jeho operačním systému a následná možnost aplikaci rozšířit. Důležitý byl z počátku také požadavek pro spuštění aplikace na webhostingu, bez nutnosti použití vlastního serveru. Tyto požadavky korespondovaly s použitím frameworku Laravel 5, který zároveň nabízel také základní bezpečnostní prvky, které byly pro aplikaci dostačující – například vlastní šifrování hesla či CSRF ochrana.

4.1.1 Webhosting a server

Pro hostování webové aplikace byl zvolen webhosting firmy Wedos, jehož parametry, jako například neomezená velikost prostoru pro webovou aplikaci, dostatečně velký prostor pro databázi a e-maily, základní antispamová kontrola či podpora používání CRON, byly dostačující pro provoz naší aplikace a zároveň nebránily v růstu celé aplikace a firmy.

S postupem času bylo rozhodnuto, že se pro hostování aplikace použije server. Bylo zvoleno řešení Cloud VPS od firmy OVH. Firma OVH byla vybrána, protože se jedná o velkou firmu a tedy úroveň služeb by mohla být vyšší než u menších firem a také díky příznivé cenové nabídce. Místo vlastního serveru byla dána přednost VPS, protože celý server by byl pro potřeby webové aplikace zbytečný. Cloudové řešení vyhrálo z toho důvodu, že by mělo zajišťovat lepší stabilitu než klasické VPS.

4.1.2 PHP a Laravel 5

Pro vytvoření aplikace byl zvolen jazyk PHP, protože aplikace, napsána v tomto jazyce, šla spustit na webhostingu a mohla být postavena na připraveném CMS, který byl napsán rovněž v tomto jazyce.

S tím, že aplikace byla postavena na již vytvořeném CMS souvisí také to, že byl použit framework Laravel 5. Ten byl zvolen proto, protože byl v době vyvíjení CMS jedním z rychlejších frameworků, měl přehlednou dokumentaci a rostla jeho popularita mezi vývojáři. Variantou byla také možnost postavení aplikace na „čistém“ PHP, což se ale jevilo jako neefektivní, a to z toho důvodu, že v případě frameworku má vývojář již velkou část funkcí připravenou a de facto na něj nabaluje již své specifické potřeby,zatímco v případě vývoje „na zelené louce“ by se muselo řešit vše od začátku. Sice by výsledná aplikace mohla být rychlejší, na druhou stranu při vývoji za pomocí frameworku se může vývojář zaměřit na již čistě konkrétní problémy, související s danou aplikací a spoustu potřebných funkcí má framework již v sobě implementovaných.

Framework byl rozšířen o tyto pluginy:

sboo/multiauth

Multiauth je plugin, který umožňuje autentifikovat uživatele v rámci více tabulek. Díky tomu lze mít klienty a administrátory ve dvou odlišných tabulkách a není nutno zasahovat do samotného procesu autentifikace. Plugin je k dispozici pod MIT licencí, což je licence, umožňující použití zdrojových kódů i v komerčních projektech, přičemž programátor může tyto zdrojové kódy libovolně modifikovat.

barryvdh/laravel-debugbar

Laravel debugbar je velmi užitečný plugin, který umožňuje vývojáři zobrazit si tzv. debug bar, ve kterém vidí spoustu užitečných informací o aplikaci a konkrétním dotazu. Za zmínku stojí například zobrazení dotazů, jež byly nad databází vykonány, včetně doby trvání. Díky tomu je vývojář schopen zjistit, kde se program zadrhává a může optimalizovat své dotazy. Této funkci sekunduje možnost zobrazení si timeline daného dotazu. Plugin je dostupný pod MIT licencí.

simplesoftwareio/simple-qrcode

Pro potřeby generování QR kódů byl zvolen plugin Simple QRCode. Plugin je dostupný pod MIT licencí.

Pomocí tohoto pluginu jsou v aplikaci generovány QR kódy s SPAYD (Short Payment Descriptor), což je řetězec, obsahující informace o platbě, jako je výše platby,

její splatnost či číslo účtu příjemce. Tento QR kód stačí naskenovat pomocí aplikace mobilního bankovnictví a veškeré údaje se automaticky předvyplní.

4.1.3 Výchozí systém

Jak již bylo naznačeno, jako základ byl použit CMS systém, běžně používán pro tvorbu webových stránek či e-shopů. Tento systém byl použit primárně z důvodu urychlení vývoje, protože měl v sobě již implementovány některé funkce a moduly, které stačilo jen lehce upravit, aby mohly být použity pro tuto aplikaci.

4.1.4 jQuery a přidružené knihovny

Pro vytvoření frontendu byla použita knihovna jQuery pro její rozšířenost, velké množství dostupných pluginů a jednoduchost použití.

S jQuery byla použita také spousta pluginů, jež řešily specifické potřeby na frontendu. Jmenovitě se jedná o tyto pluginy:

autoNumeric.js

jQuery plugin, využíván pro formátování inputů, do nichž píšou uživatelé částky. Při psaní tak uživatel vidí hezky naformátované číslo a je to pro něj čitelnější. Plugin je dostupný pod MIT licencí.

DataTables

Užitečný jQuery plugin, který je schopen načtenou HTML tabulku změnit v dynamickou tabulku. Data do této tabulky mohou být vložena v HTML nebo načtena samotným pluginem pomocí AJAXu.

V tabulce lze poté hledat podle sloupečků, řadit data podle zvoleného sloupečku, rozdělit výsledky do více stran, zvolit si počet záznamů na stránku nebo používat vnořené tabulky – například u výpisu zákazníků je tlačítko, které rozevře vnořenou tabulku, v níž jsou vypsány investice daného uživatele.

Velkou výhodou je také fakt, že lze data z tabulky exportovat do PDF, CSV či odeslat přímo na tisk. Plugin je dostupný pod MIT licencí.

DatePicker

Pro potřeby zadávání datumů ze strany administrátorů byl do aplikace implementován plugin DatePicker. Plugin nahradí input pro zadávání datumu tři selecty, skrze kterých si lze vybrat den, měsíc a rok. Tento plugin jsem vytvořil pro použití v této aplikaci.

Flatpickr

Jedná se o jednoduchý plugin, který umí k inputu připojit kalendář, ze kterého si uživatel může vybrat datum. Tento plugin je využíván, pro usnadnění zadávání datumů a zároveň je zajištěn jednotný formát těchto dat. Nabízí široké možnosti nastavení a podporuje několik jazykových mutací. Plugin je dostupný pod MIT licencí.

Fullcalendar

Fullcalendar je plugin, použitý pro zobrazování kalendáře v aplikaci. Kalendář je interaktivní a lze v něm přepínat mezi různými pohledy – den, měsíc nebo agenda Uživatel se v něm může posouvat dopředu či dozadu vůči aktuálnímu datu. Umožňuje nastavení callbacků, díky kterým se dají do kalendáře jednoduše implementovat možnosti pro vytváření, editaci a mazání událostí.

Plugin umí data načítat AJAXem, s využitím lazyloadingu. Díky tomu jsou načtena vždy pouze potřebná data.

Plugin je opět dostupný pod MIT licencí.

Chart.js

Chart.js je plugin pro vykreslování grafů, dostupný pod MIT licencí. Nabízí široké možnosti nastavení grafů, výběr z několika druhů grafů, kvalitní dokumentaci a také možnosti customizace, jako například přepsání metod pro zobrazování tooltipů. Je postaven na HTML5 a využívá pro vykreslování grafů canvas. Plugin podporuje responzivní vykreslování grafů.

jQuery File Upload

Tento plugin se stará o zprovoznění možnosti nahrávání souborů pomocí AJAXu. Standardně nelze skrze AJAX soubory nahrávat, respektive odesílat ve formuláři, z tohoto důvodu byl použit plugin, který tuto funkcionalitu „doplňuje“. Plugin umožňuje zobrazovat progress bar pro zobrazení nahrávání souborů a vývojáři nabízí různé callbacky.

Plugin je dostupný pod MIT licencí. Pro spuštění tohoto pluginu bylo nutno přidat také jQuery UI, rovněž dostupné pod MIT licencí.

OrgChart

OrgChart plugin umožňuje jednoduše vykreslovat stromové struktury v aplikaci. Využíván je primárně pro zobrazení struktury obchodních zástupců společnosti. Plugin je dostupný pod MIT licencí.

StyleSelect

Jedná se o plugin, díky kterému lze přestylovat selecty do designu, jež koresponduje s designem webové aplikace. Funguje tak, že původní select schová a vytvoří si vlastní dropdown, ve kterém jsou vidět jednotlivé možnosti, dostupné původně v selectu. Plugin umožňuje také hledání mezi jednotlivými možnostmi. Tento plugin jsem si vytvářel sám.

Swiper.js

Pro potřeby zobrazení příspěvků na nástěnce byl zvolen plugin Swiper.js. Díky němu stačí do HTML do připravené kostry vložit zvolené příspěvky a plugin z nich udělá rotovací galerii. Plugin je dostupný pod MIT licencí.

Toogle

Plugin Toggle využívám pro přepínání záložek, rozevírání různých částí webové stránky či otevírání modalů, ať už předpřipravených v HTML nebo dynamicky načtených skrze AJAX. Plugin jsem si vytvářel pro vlastní potřeby, protože jsem potřeboval řešení, které bych si mohl volně a jednoduše modifikovat.

jQuery Validation Plugin

Jedná se o jeden z nejznámějších jQuery pluginů, řešící problematiku realtime validací. U pluginu lze velmi jednoduchým způsobem nadefinovat pravidla pro jednotlivé inputy – zda je nutné jej vyplnit, zda se má jednat o e-mail, jaká má být minimální délka řetězce, zda si mají dva inputy odpovídat (například heslo a ověření hesla) atd. Navíc poskytuje vývojáři možnost dopsat si jednoduchým způsobem vlastní pravidla. Plugin je dostupný pod MIT licencí.

4.2 Postup při vývoji první verze aplikace

Při vývoji této webové aplikace bylo použito plánování úkolů v rámci týdenních sprintů, kterým sekundovaly schůzky s klientem.

Vzhledem k tomu, že zpočátku se jednalo o vcelku triviální aplikaci, nebylo třeba s klientem zdlouhavě diskutovat o parametrech aplikace a vše se stihlo relativně rychle. Jednotlivé fáze tvorby prvotní verze aplikace bych rozdělil následovně:

1. První schůzka s klientem, seznámení se s jeho potřebami, diskuze nad navrhnutým řešením.

Protože byly již dopředu známy základní požadavky na aplikaci a zároveň měla být aplikace z počátku jednoduchá, během první schůzky se povedlo ujasnit, co se od aplikace očekává a byl klientovi prezentován návrh řešení. Velkou výhodou je fakt, že klient má zkušenosti s IT, a proto zvládá velmi dobře formulovat své požadavky.

2. Tvorba grafického návrhu

Po vyjasnění si všech požadavků byl zpracován grafický návrh webové aplikace. Díky tomu, že byla ze strany klienta k dispozici vcelku jasná představa ohledně designu aplikace, zvládl se celý design vytvořit a vyladit během dvou iterací – byl vytvořen první návrh, který byl zaslán klientovi a na základě jeho připomínek byla vytvořena finální verze grafiky.

3. Vytvoření HTML šablon

Tvorba HTML šablon neskýtala žádné záludnosti ani zvláštní komplikace. V HTML se nakódovaly šablony, do kterých byly přidány některé JavaScriptové

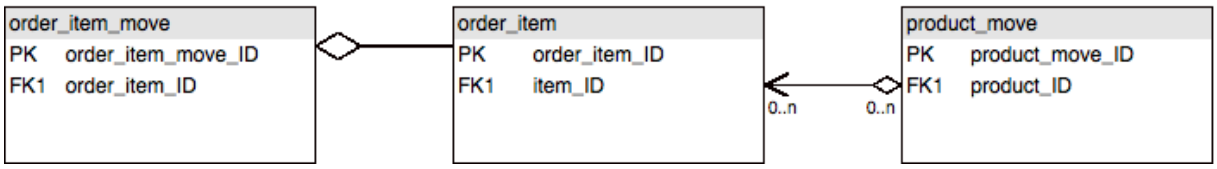
funkcionality, jako například rolovací text pro zobrazení hodnot akcií v boxu nebo validace formulářů.

Šablony byly vytvořeny tak, aby byly mezi sebou vzájemně propojeny a klient tak měl při jejich procházení pocit, že již před sebou vidí reálnou aplikaci. Díky tomu šlo odhalit nesrovnalosti ve fungování či pohybování se v rámci aplikace, ještě než se začaly šablony nasazovat na systém. V rámci šablon nebyla potřeba provádět velké zásahy, a proto se mohlo vcelku brzo pracovat na napojení na backend.

4. Napojení na backend a jeho úpravy

Protože měla být aplikace vcelku jednoduchá, byl použit již připravený CMS systém. V něm bylo upraveno pár modulů - například modul pro objednávky se upravil tak, aby mohl být použit pro investice. Dále bylo třeba doprogramovat určité funkcionality, mezi které patřilo stahování kurzů zvolených akcií, jež se potom zobrazovaly uživatelům na nástěnce.

Do systému byla potřeba dle požadavků klienta implementovat hromadné zadávání pohybů k investicím. Při implementaci tohoto požadavku bylo nutné vzít také v potaz to, že každá investice může mít ještě své vlastní pohyby jako jsou vklady či výběry. Z tohoto důvodu bylo nakonec zvoleno řešení, kde se pro ukládání pohybů používají dvě tabulky. První tabulka s názvem `product_move` v sobě obsahuje informace o pohybech jednotlivých produktů. Druhá tabulka, nazývající se `order_item_move`, obsahuje pohyby, které se týkají konkrétní investice.



Obrázek 3 Relace mezi order_item, order_item_move a product_move (zdroj: Vlastní)

Nyní bylo nutné vyřešit, jak počítat aktuální stav investice. Přepočítávání její hodnoty vždy, když si klient investici zobrazí, se nejevilo jako ideální řešení, proto se k investici vytvořil atribut, který se nazval `present_amount`. Hodnota tohoto atributu byla automaticky počítána vždy, když klient zadal nové pohyby, nebo když se vložil nový pohyb u investice. Protože byly potřebné hodnoty ve dvou tabulkách, bylo nutné vytvořit

dotaz na databázi, který by vrátil výsledky z těchto dvou tabulek najednou. Výsledný dotaz má tuto podobu:

```
SELECT tall.move,
       tall.type,
       tall.date,
       tall.created_at,
       tall.move_id
FROM   ((SELECT order_item_move.order_item_move_id AS move_ID,
                order_item_move.move,
                order_item_move.type,
                order_item_move.date,
                order_item_move.created_at
         FROM   order_item_move
         WHERE  order_item_move.date >= DATUM_AKTIVACE_INVESTICE
                AND order_item_move.order_item_id = ID_DANE_INVESTICE)
        UNION ALL
        (SELECT product_move.product_move_id AS move_ID,
                product_move.move,
                product_move.type,
                product_move.date,
                product_move.created_at
         FROM   product_move
         WHERE  product_move.date >= DATUM_AKTIVACE_INVESTICE
                AND product_move.product_id = ID_PRODUKTU)) AS tall
ORDER  BY tall.date ASC,
          tall.created_at ASC
```

Zdrojový kód 6 Dotaz na databázi pro získání pohybů u produktů a investic

Tučně zvýrazněný text v dotazu popisuje, jaká hodnota byla na toto místo doplněna. V případě, že je investice již smazána nebo skončila, je do dotazu přidána ještě podmínka, která omezuje datum, aby nebylo vyšší než datum smazání nebo ukončení investice.

Během úprav celého systému byly průběžně prováděny testy jednotlivých funkcionalit, proto při nasazení stačilo věnovat otestování vcelku málo času a aplikace mohla být prezentována klientovi. Díky jeho schopnosti dobře formulovat požadavky se povedlo vytvořit aplikaci, ke které neměl prakticky žádné výtky a aplikaci mohli začít používat.

Díky tomu, že klientova společnost rostla, narůstaly také požadavky klienta na samotný systém. Některým z nich se budu věnovat v následujících kapitolách.

4.3 Implementace struktury obchodních zástupců

Jedním z prvních navazujících požadavků ze strany klienta byla implementace struktury obchodních zástupců. Šlo de facto o to, aby bylo ve struktuře vidět, který obchodní zástupce je nad kterým, tj. kdo koho přivedl. Jednalo se o vcelku triviální úpravu, která požadovala pouze přidělat do tabulky `customer` dva sloupce – `owner_ID` a `owner_type`. Bylo nutné použít dva sloupce z toho důvodu, že do systému mají přístup dva typy účtů, tzv. user a customer, neboli obchodní zástupce. Proto bylo nutné, kromě sloupce owner_ID, který odkazuje skrze primární klíč usera nebo customera na příslušného usera či customera, přidat ještě sloupec, v němž data mohou nabývat hodnot USER a CUSTOMER a odlišuje, o který typ uživatele se jedná.

4.4 Implementace provizního systému

Na implementaci struktury obchodních zástupců navazovala implementace provizního systému. Tento systém procházel mnoha modifikacemi, nicméně nyní funguje na tomto principu – jsou čtyři druhy obchodních zástupců – obchodní zástupce, top obchodní zástupce, manažer a top manažer. Typ obchodního zástupce se nastavuje skrze office a určení typu se řídí interními firemními předpisy. Na základě typu obchodního zástupce a objemu portfolia, jež spravuje, se určuje výše provize, kterou dotyčný dostane.

Při rozdělování provize hraje roli také pozice obchodního zástupce v rámci struktury. Suma těchto provizí pro jednu investici se rovná 1 % z celkové hodnoty investice. Toto procento je rozděleno mezi obchodní zástupce, kteří se váží k dané investici, a to tak, že se zjistí, který obchodní zástupce je „vlastníkem“ konkrétní investice, a poté se od tohoto obchodního zástupce postupuje nahoru přes jednotlivé úrovně. Tedy pokud je obchodní zástupce na nejnižší pozici, najde se v jeho struktuře obchodní zástupce o úroveň výše a tomu se přidělí určitý podíl z provize. Obdobným způsobem se pokračuje, dokud se nedojde k top manažerovi.

Z počátku se určoval obchodní zástupce pro danou investici tak, že každý klient má uloženo jeho id pod položkou owner_ID. Postupem času ovšem nastal problém, kdy bylo třeba umožnit, aby jeden klient měl investici od více obchodních zástupců. Za tímto účelem se u investice vytvořily opět sloupce owner_ID a owner_type, do nichž se ukládá odkaz na daného obchodního zástupce.

Pro zjištění, jakou má dostat obchodní zástupce dané investice provizi je použita metoda getOwnerProfit.

```
public function getOwnerProfit($owner) {
      $ownerPortfolioAmount = $owner->getCustomersDatePortfolioAmount(
                                                  null,
                                                  $this->date);
      $ownerProfit          = null;
      switch ($owner->type) {
      case 'REP':
            if ($ownerPortfolioAmount->own < 5000000) {
                  $ownerProfit = 0.005;
            } else {
                  $ownerProfit = 0.006;
            }
            break;
      case 'TOP_REP':
            if ($ownerPortfolioAmount->total < 10000000) {
                  $ownerProfit = 0.007;
            } else {
                  $ownerProfit = 0.008;
            }
            break;
      case 'REP_MANAGER':
            $ownerProfit = 0.009;
            break;
      case 'TOP_REP_MANAGER':
            $ownerProfit = 0.01;
            break;
      }
      return $ownerProfit;
}
```

Zdrojový kód 7 Metoda getOwnerProfit

V této metodě na základě typu obchodního zástupce a výše jeho portfolia se určuje, jakou má obchodní zástupce dostat provizi.

Na tuto metodu navazuje metoda, jež se nazývá getStructProfits.

```
public function getStructProfits($owner, $ownerProfit) {
      $profits                          = new \stdClass();
      $profits->ownerTopRepProfit       = 0;
      $profits->ownerRepManagerProfit   = 0;
      $profits->ownerTopRepManagerProfit = 0;
      switch ($owner->type) {
      case 'REP':
            // owner top rep
            $profits->ownerTopRep = $owner->getTopRep();
            if ($profits->ownerTopRep != null) {
                  $profits->ownerTopRepProfit =
                              0.007 - $ownerProfit;
            }
            // owner rep manager
            $profits->ownerRepManager = $owner->getRepManager();
            if ($profits->ownerRepManager != null) {
                  $profits->ownerRepManagerProfit = 0.009 -
                              ($ownerProfit +
                                    $profits->ownerTopRepProfit);
            }
            // owner top rep manager
            $profits->ownerTopRepManager = $owner->getTopRepManager();
            if ($profits->ownerTopRepManager != null) {
                  $profits->ownerTopRepManagerProfit = 0.01 -
                              ($ownerProfit +
                              $profits->ownerTopRepProfit +
                              $profits->ownerRepManagerProfit);
            }
            break;
      case 'TOP_REP':
            // owner rep manager
            $profits->ownerRepManager = $owner->getRepManager();
            if ($profits->ownerRepManager != null) {
                  $profits->ownerRepManagerProfit = 0.009 -
                              ($ownerProfit);
            }
```

```
            // owner top rep manager
            $profits->ownerTopRepManager = $owner->getTopRepManager();
            if ($profits->ownerTopRepManager != null) {
                  $profits->ownerTopRepManagerProfit = 0.01 -
                              ($ownerProfit +
                              $profits->ownerRepManagerProfit);
            }
            break;
      case 'REP_MANAGER':
            // owner top rep manager
            $profits->ownerTopRepManager = $owner->getTopRepManager();
            if ($profits->ownerTopRepManager != null) {
                  $profits->ownerTopRepManagerProfit = 0.01 -
                              ($ownerProfit);
            }
            break;
      }
      return $profits;
}
```

Zdrojový kód 8 Metoda getStructProfits

Tato metoda na základě typu obchodního zástupce dopočítá provize pro zbývající obchodní zástupce a vrátí tyto hodnoty.

Po zavolání těchto metod jsou k dispozici již všechny potřebné informace o provizích, a proto může být zavolána metoda processStructProfits, která uloží u jednotlivých obchodních zástupců příslušné provize.

```
public function processStructProfits($investment, $owner,
$ownerProfit) {
      $profits = $this->getStructProfits($owner, $ownerProfit);
      // add profits
      if ($profits->ownerTopRepProfit != 0) {
            $this->addUserProfit($investment,
                                    $profits->ownerTopRepProfit,
                                    $profits->ownerTopRep);
      }
      if ($profits->ownerRepManagerProfit != 0) {
            $this->addUserProfit($investment,
                                    $profits->ownerRepManagerProfit,
                                    $profits->ownerRepManager);
      }
      if ($profits->ownerTopRepManagerProfit != 0) {
            $this->addUserProfit($investment,
                                    $profits->ownerTopRepManagerProfit,
                                    $profits->ownerTopRepManager);
      }
}
```

Zdrojový kód 9 Metoda processStructProfits

Metoda addUserProfit v sobě obsahuje uložení všech potřebných informací k danému obchodnímu zástupci do pole. Po ukončení cyklu, v němž se projdou všechny investice, je vytvořeno pole s jednotlivými obchodními zástupci, kde jsou uloženy také investice, z nichž mají provize i výše těchto provizí. Toto pole se prochází v cyklu a pro každého obchodního zástupce se generuje PDF dokument, obsahující informace o provizích. Tento dokument včetně doplňujících údajů vidí každý obchodní zástupce ve svém office. Jakmile s tímto dokumentem souhlasí – tj. odškrtne, že je vše v pořádku, provede se kontrola, zda má tento obchodní zástupce nastaveno, že se mají jeho provize posílat na bankovní účet nebo připsat do jím zvolené investice. Pokud má zvolenou investici, tyto provize jsou do investice připočteny, v opačném případě je v systému pod profilem správce vidět čekající výplata s vygenerovaným QR kódem pro odeslání platby. Ten stačí naskenovat mobilní aplikací a odeslat. Jakmile je toto provedeno, uživatel odškrtne, že byly prostředky odeslány.

4.5 Automatizované připisování plateb na základě e-mailů z banky

Vzhledem k tomu, že se začínali hromadit klienti a investice, stávalo se komplikovaným hlídat vklady do investic, jež klienti zasílali na bankovní účet. Z tohoto důvodu byla do systému implementována automatická kontrola vkladů na bankovní účet. Požadavkem bylo, aby bankovní účet zůstal u ČSOB, která v současné době nenabízela API, skrze kterou bychom mohli tento problém řešit. Proto bylo nutné pracovat s e-maily – banka zasílá v určitých intervalech na zvolený e-mail informace o příchozích platbách. E-maily chodí na server, kde běží také samotná aplikace, proto se nastavil CRON, který každou hodinu spustí PHP skript, jež přečte nové e-maily a na základě nich poté páruje platby k daným investicím.

Celý proces je realizován souborem CSOBEmailNotifications.php, který v sobě obsahuje všechny potřebné metody.

Metoda `processReceivedPayments` je metodou, která v sobě obsahuje volání dalších tří metod. Jedná se o metodu, díky které se provedou všechny potřebné úkony. Volané metody v sobě obsahují kontrolu připojení pro čtení e-mailů, následné čtení těchto e-mailů a posléze zase ukončení připojení.

Metoda, jež čte e-maily se nazývá `loopEmails`. Tato metoda v sobě obsahuje cyklus, ve kterém se prochází jednotlivé e-maily a následné volání metody `log`, která na zvolený e-mail pošle informace o provedených platbách a zapíše informace do souboru.

```
$emailsCount = imap_num_msg($this->emailConnection);
$processedPayments = [];
for ($msgNo = 1; $msgNo <= $emailsCount; $msgNo++) {
      $msg = new CSOBEmail($this->emailConnection, $msgNo, $this);
      if ($msg->isOk()) {
            foreach ($msg->getPayments() as $payment) {
                  $payment->processPayment();
                  $processedPayments[] = $payment;
            }
            $msg->setFlagged();
      }
}
```

Zdrojový kód 10 Cyklus procházení e-mailů

Princip procházení spočítá v tom, že se z připojení získá, kolik je celkem e-mailů a následně se v cyklu `for` prochází čísla od jedné až do počtu e-mailů, včetně. V tomto cyklu se vytvoří instance třídy CSOBEmail, která pomáhá pracovat s daným e-mailem. Tato třída v sobě obsahuje metodu `isOk`, která slouží ke kontrole, zda může být daný e-mail zpracován. V této metodě se primárně kontroluje to, zda byl již e-mail předtím přečten nebo ne. Mechanismus kontroly je řešen tak, že je e-mail po přečtení označen za „flagged“. E-maily, které jsou takto označeny jsou poté přeskakovány a nezpracovány.

Další metodou je metoda `getPayments`. Tato metoda slouží k tomu, aby se e-mail rozdělil na jednotlivé platby a ty mohly být dále zpracovány.

```
public function getPayments() {
      $return = [];
      $paymentsArray = [];
      if (strpos($this->body, 'zaúčtovaná transakce') !== false) {
            $paymentsArray = explode('zaúčtovaná transakce',
                                          $this->body);
            unset($paymentsArray[0]); // remove first part
      } else if (strpos($this->body,
                              'zaúčtován hotovostní vklad') !== false)
{
            $paymentsArray = explode('zaúčtován hotovostní vklad',
                                          $this->body);
            unset($paymentsArray[0]); // remove first part
      }
      foreach ($paymentsArray as $payment) {
            // remove last part from payment string
            $paymentArray  = explode('Zůstatek na účtu po zaúčtování
                                          transakce:', $payment);
            $paymentString = $paymentArray[0];
            $return[]      = new CSOBEmailPayment($paymentString,
                                                      $this->date);
      }
      return $return;
}
```

Zdrojový kód 11 Metoda getPayments

Princip rozdělení e-mailu na jednotlivé platby spočívá v tom, že e-mail má vždy jednotný formát a každá informace o platbě začíná textem „zaúčtovaná transakce“ nebo „zaúčtován hotovostní vklad“. V metodě je tedy provedena kontrola, zda tělo zprávy

obsahuje jeden z řetězců a následně je tělo podle tohoto řetězce rozděleno na podřetězce, jež obsahují informace o jednotlivých platbách. Pro snadnější práci s platbou je použita instance třídy CSOBEmailPayment, nad kterou se poté mohou volat připravené metody, jako například `getVS`, `getAmount` apod.

4.6 Implementace dvou faktorového ověřování

Z důvodu zajištění větší bezpečnosti přístupu do aplikace se nyní pracuje na implementaci dvou faktorového ověřování, které funguje na takovém principu , že po zadání kupříkladu přihlašovacích údajů je uživatel vyzván, aby zadal například kód, který mu přijde SMSkou nebo kód, jež najde v aplikaci, což je náš případ.

Pro dvou faktorové ověřování je využívána aplikace Google Authenticator. Celý proces ověřování funguje tak, že u uživatele, který si dvou faktorové ověřování nastavil, je uložen jedinečný kód. Tento kód si při nastavování vloží nebo naskenuje uživatel do mobilní aplikace.

Na základě tohoto kódu se ve zvolených časových intervalech generuje nový ověřovací kód, který uživatel z mobilní aplikace zadá při přihlašování. Webová aplikace tento kód ověří a v případě shody proběhne přihlášení uživatele.

Pro implementaci tohoto dvou faktorového ověřování byl použit již vytvořený kód z balíčku PHPGangsta/GoogleAuthenticator, který má již implementovány všechny potřebné metody. Proces ověřování poté vypadá ve zjednodušené podobě následovně:

```
if ($gaCode != '') {
      $googleAuthenticator = new GoogleAuthenticator();
      $checkResult         = $googleAuthenticator->verifyCode(
                                          $activeGASecret->secret,
                                          $gaCode,
                                          2);
}
```

Zdrojový kód 12 Ověření správnosti u 2 faktorového ověřování

V proměnné `$gaCode` je uložen ověřovací kód, jež uživatel zadal při odeslání formuláře. V proměnné `$activeGASecret` je uložen aktivní kód, jež uživatel naskenoval při aktivaci dvou faktorového ověřování. Metoda `verifyCode` vrací hodnotu `true` nebo `false`, podle toho, zda byl kód uživatelem zadán správně.

4.7 Napojení na pool pro těžbu kryptoměn

Protože klient rozšířil nabízené produkty, muselo se řešit napojení webové aplikace na pool pro těžbu, a to ze dvou důvodů:

- Automatické připisování zisku z těžby klientům do jejich investic.
- Zobrazení statistik těžební farmy přímo v office.

Napojení je realizováno skrze API, na které se volá každých 15 minut dotaz. Primární data, jež se z poolu získávají jsou tato:

- Historie poolu – tato data v sobě obsahují hlavně informace o výkonu dané farmy.
- Výplaty poolu – tato data v sobě obsahují informace o tom, jaké odměny obdržela farma z poolu.

Na základě těchto dat je možno v aplikaci zobrazovat statistiky dané farmy a také vypočítávat výplaty klientům do jejich investic.

Tento výpočet se provádí každý den v 01:15. Během výpočtu se získá průměrný výkon farmy za posledních 24 hodin a seznam nevyplacených výplat – tj. výplaty, které pool farmě vyplatil ale systém je ještě nezpracoval.

Tyto výplaty se prochází v cyklu a u každé výplaty se spočte částka, která připadne konkrétní investici. Tato částka se vypočte tak, že se spočte výše výplaty pro určitou jednotku výkonu a tato hodnota se vynásobí výkonem, jež si klient v dané investici objednal.

5 Závěr

Cílem této bakalářské práce bylo vytvoření webové aplikace, jež usnadní firmě správu klientů, investic, zjednoduší administrativu a zjednoduší některé aspekty, týkající se fungování firmy. Byť byl tento cíl naplněn, na aplikaci stále probíhají úpravy a vylepšování, aby co nejvíce odpovídala potřebám firmy. Mezi tyto úpravy patří například napojení na pool, implementace dvou faktorového ověřování, třeba vytvoření nového typu uživatelského profilu, jež může nahlížet na celou firmu, ale má omezené funkce nebo nyní probíhající rozdělení aplikace na dvě různé aplikace, kdy jedna bude sloužit primárně pro vedení společnosti, pro správu investic, produktů, zadávání pohybů, či generování affiliate reportů a druhá aplikace bude pro samotné klienty a obchodní zástupce.

5.1 Přínosy pro společnost

Dle vyjádření klienta soudím, že je s aplikací spokojený a ušetřila mu velké množství času. Čas, který klient díky této aplikaci ušetřil, je komplikované stanovit, nicméně pro představu bych si dovolil provést odhad.

Pro provedení odhadu uspořeného času je třeba zavést určité hodnoty, s nimiž budu počítat. V tabulce níže je hodinové vyjádření času, potřebného k určitým úkonům.

Úkon	Staré řešení	Nové řešení
Čas potřebný k vytvoření investice (v hod.)	0,5	0,083
Prům. čas potřebný k zadání pohybů u investic (hod.)	0,3	0,083
Prům. čas potřebný k hromadnému výpočtu provizí (hod.)	0,3	0,083
Prům. čas potřebný k zaevidování provize u OZ (hod.)	0,25	0

Tabulka 1 Časové náročnosti některých úkonů

Na základě těchto údajů jsem vypočítal hodnoty časové náročnosti pro některé úkony a také výpočet úspory času (viz Tabulka 2). Hodnoty některých řádků tabulky bych rád okomentoval a vysvětlil, jakým způsobem jsem k nim dospěl.

Počet investic, počet klientů, počet obchodních zástupců

Hodnoty pro 1. rok jsem získal z databáze ke dni 1.4.2017, hodnoty pro 2. rok jsou z databáze k datu 16.4.2017. Hodnoty pro další rok jsem vypočítal jako součet hodnoty předchozího roku a hodnoty rozdílu mezi dvěma posledními lety (tj. přírůstek),

který jsem vynásobil ½. Tímto jsem se pokusil simulovat růst těchto stavů s přihlédnutím k slábnutí růstu. Hodnoty jsou zaokrouhleny na celá čísla dolů.

Vytvoření investic

Čas potřebný k vytvoření investic jsem vypočítal na základě vzorce $Počet\ investic\ .Vytvoření\ investic$.

Zadávání pohybů

Tuto hodnotu jsem získal na základě vynásobení doby, nutné pro zadávání pohybů s průměrným počtem pracovních dnů v roce. Při těchto výpočtech jsem pracoval s číslem 250.

Výpočet provizí

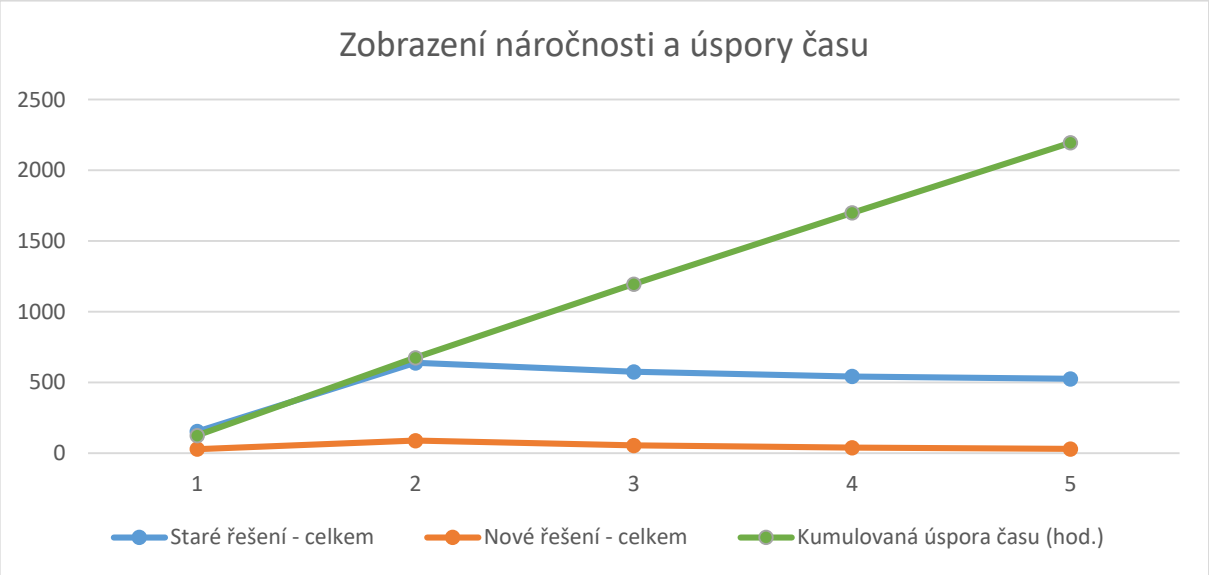
Výpočet provizí byl založen na součtu dvou hodnot, a to času potřebném k hromadnému výpočtu provizí a připočtení času, potřebného k zaevidování provizí pro každého obchodního zástupce zvlášť. Protože se počet obchodních zástupců v čase měnil, pokusil jsem se počet obchodních zástupců odhadnout, a to tak, že jsem k předchozímu stavu obchodních zástupců připočetl ½ nových obchodních zástupců pro aktuální rok. Tj. v 1. roce bylo 23 obchodních zástupců. Ve 2. roce bylo obchodních zástupců 84, přírůstek tedy činil 61 obchodních zástupců, polovina z tohoto čísla je cca 30,5. Pro daný rok jsem tedy počítal s číslem 53,5.

V posledních řádcích tabulky je vidět úspora času a kumulovaná úspora času. Vidíme, že je roční úspora pro klienta značná a kumulovaná úspora se v průběhu 5 let vyšplhala až na téměř 2200 hodin, což je více než 250 pracovních dní. Zároveň je potřeba u tohoto odhadu brát v potaz, že se počítalo s klesajícím nárůstem počtu investic, klientů a obchodních zástupců. Pokud by tento nárůst byl větší, úspory klienta budou ještě markantnější. Ve výpočtech také není zahrnut čas, potřebný k další administrativě, hledání klientů či potřebných údajů.

Název hodnoty	1. rok	2. rok	3. rok	4. rok	5. rok
Stavy					
Počet investic	79	878	1277	1476	1575
Počet klientů	79	805	1168	1349	1439
Počet obchodních zástupců	23	84	114	129	136
Staré řešení					
Vytvoření investic	39,500	399,500	199,500	99,500	49,500
Zadávání pohybů	75	75	75	75	75
Výpočet provizí	38,100	164,100	300,600	368,100	401,100
Staré řešení - celkem	152,600	638,600	575,100	542,600	525,600
Nové řešení					
Vytvoření investic	6,557	66,317	33,117	16,517	8,217
Zadávání pohybů	20,750	20,705	20,750	20,750	20,750
Výpočet provizí	0,996	0,996	0,996	0,996	0,996
Nové řešení - celkem	28,303	88,063	54,863	38,263	29,963
Shrnutí					
Úspora času (hod.)	124,297	550,537	520,237	504,337	495,637
Kumulovaná úspora času (hod.)	124,297	674,834	1195,071	1699,408	2195,045

Tabulka 2 Zobrazení časů potřebných pro některé úkony, výpočet úspory

K tabulce přikládám grafické znázornění odhadované časové úspory.



Obrázek 4 Grafické znázornění časové náročnosti pro některé úkony a úspory času (zdroj: Vlastní)

5.2 Ohlas uživatelů

Pro zjištění ohlasů uživatelů webové aplikace jsme rozeslali dotazník deseti uživatelům webové aplikace. V dotazníku bylo celkem osm otázek, pět z nich s výběrem

jedné či více možností a tři otázky s otevřenou odpovědí. Znění otázek, odpovědí, včetně jejich grafického znázornění je vidět v Příloze 1 - Dotazník.

Je pozitivní, že uživatelé nezaznamenávají častý výskyt chyb a jedná se tak spíše o občasný jev. To je, dle mého názoru pro aplikaci, která byla vytvořena jediným člověkem, a která obsahuje dost funkcí a používá ji relativně velké množství lidí, velmi dobrý výsledek.

Uživatelé také kladně ohodnotili rychlost opravy případných chyb. Odpovědi u otázky, týkající se rychlosti implementace požadavků uživatelů si myslím, že byly ovlivněny částečně tím, že mohou uživatelé považovat implementaci požadavku a opravu chyby za jednu a tutéž záležitost, což se podepsalo na jejich hodnocení. Na druhou stranu je pravda, že velkou část požadavků se snažím řešit v co nejkratším možném termínu.

Při pohledu na otevřené otázky, co se uživatelům na office líbí a nelíbí, vidíme, že nejčastěji se v pozitivních hodnoceních vyskytuje odpověď, že je office přehledný, jednoduchý a má přívětivé prostředí. Při pohledu do negativních ohlasů vidíme, že bylo dvakrát zmíněno, že je office pro uživatele moc tmavý. Máme tedy dvě skupiny uživatelů, jedné se grafické prostředí líbí a druhá by uvítala zpět světlou variantu. Z tohoto důvodu budu do office implementovat možnost, aby si uživatel mohl zvolit mezi světlou a tmavou verzí.

Poslední otázka nabízela uživatelům možnost, vyjádřit svůj názor, co by do office rádi přidali. Z této otázky jsou pro mě nejdůležitější čtyři odpovědi vyjadřující názory, které mě mohou navést v budoucím vývoji aplikace. Co se týče odpovědi číslo 3 s proklikem do e-mailu, to bylo nejspíš způsobeno tím, že se obchodní zástupce přihlásil pod druhým účtem, jež nemá toto tlačítko vidět. Co se týče zbývající třech podnětů, některé z nich jsou již ve zpracování a další na základě tohoto ohlasu plánuji do systému implementovat.

Z celkového náhledu na odpovědi v dotazníku můžeme konstatovat, že jsou uživatelé se samotnou aplikací spokojeni a aplikace jim usnadňuje jejich práci, což byl primární cíl vytvoření této aplikace a také bakalářské práce.

5.3 Zhodnocení zvoleného postupu

Při zpětném ohlédnutím se za celým procesem vývoje této aplikace neshledávám, že bych při vývoji aplikace provedl některé kroky, které by mi samotný vývoj nějak výrazně komplikovaly nebo působily do budoucna potíže.

Dle mého názoru tomuto relativně hladkému průběhu vývoje aplikace vděčím z velké části tomu, že požadavky na aplikaci rostly postupně s klientem a vývoj aplikace probíhal pozvolna od velmi jednoduché aplikace až po výslednou, vcelku komplexní, aplikaci.

Vzhledem k velikosti samotné aplikace bych nyní, při zpětném ohlédnutí, volil řešení, kdy bych aplikaci postavil na čisté verzi frameworku Laravel 5 a nepoužil bych již připravený CMS systém, jež běžně používám. Na druhou stranu dle původního zadání bylo řešení postavené na CMS systému dostačující a ani nyní ničemu příliš nevadí.

5.4 Budoucí vývoj

Z hlediska budoucího vývoje této aplikace je nutné provádět její průběžné úpravy dle požadavků uživatelů a opravovat chyby, které byly v aplikaci objeveny.

Aplikace bude také procházet postupnými úpravami dle požadavků klienta. V současnosti již probíhá jedna z iterací úprav, která má za cíl vydání nové verze office, jež bude umět validovat formáty českých, polských a slovenských rodných čísel a na jejich základě zároveň uživateli předvyplní také datum narození.

Další významnou úpravou, jež se chystá jsou e-mailové notifikace. Tato úprava částečně implementuje na požadavek z dotazníku na upozornění vkladů a výběru prostředků. Nyní jsem do systému implementoval systém pro logování prakticky veškerých aktivit, jež uživatelé v systému provedou, například přihlášení, změna údajů, zaevidování dokumentu na pobočce, žádost o výběr investice. Toto logování si nyní může prohlížet a filtrovat klient v sekci pro správu. Na toto logování nyní připravuji napojení e-mailů, díky kterému bude obchodní zástupce například informován o tom, že jeho klient zažádal o výběr z investice nebo že dokument přišel na pobočku.

Z důvodu zajištění lepší bezpečnosti bude probíhat také implementace bezpečnostního odhlášení z webové aplikace. To znamená, že každý uživatel bude mít

limit pro nečinnost, který je nastaven na pět minut. Pokud v systému neprovede žádnou akci, je vyzván, aby limit kliknutím na tlačítko prodloužil, nebo je automaticky odhlášen. Jedinou výjimkou z hlediska časového limitu bude sekretářka, jež bude mít během pracovní doby pro určitou IP adresu nastaven delší časový limit.

Na úpravy bezpečnosti navazuje také implementace nových funkcí kvůli GDPR, jež vstoupí v platnost ke konci měsíce května. V souladu s tímto nařízením bude potřeba do systému implementovat další funkce, díky kterým si bude moci uživatel zažádat o výpis všech údajů, které o něm v systému evidujeme, bude moci požádat o smazání těchto údajů a systém bude umět logovat tuto komunikaci s klientem a potvrzení jeho rozhodnutí.

S přihlédnutím k ohlasům z dotazníku je v plánu také přidělání nového barevného motivu office, s nímž přibude v office také možnost zobrazení měsíčního kalendáře a vytvoření tzv. rychlých odkazů, které uvidí uživatel na nástěnce a budou obsahovat mimo jiné odkaz na vytvoření klienta. Vytváření přehlednějších reportů je již také v plánu, bohužel zatím ale tento úkol nemá takovou prioritu jako například bezpečnost, a proto jej budu řešit až v rámci několika nejbližších měsíců.

Do budoucna také plánuji provedení refactoringu celé aplikace a zpřehlednění jejího kódu. Vzhledem k tomu, že byla aplikace původně postavena na připraveném CMS, obsahuje v sobě soubory, jež nejsou potřebné k jejímu fungování. Tyto soubory ničemu nevadí, nicméně rád bych je při budoucím refactoringu odstranil. Druhou možností je přepsání aplikace na novější verzi frameworku Laravel, která bude rychlejší a bude v sobě obsahovat pouze nutné soubory. Kterou možnost zvolíme, o tom rozhodneme před prováděním těchto změn na základě časové vytíženosti a priorit jednotlivých úkolů.

V souladu s ohlasy klienta a daty, jež mám k dispozici, usuzuji, že webová aplikace plní svůj účel, klientovi usnadňuje práci a cíl bakalářské práce byl naplněn v plném rozsahu.

Seznam použité literatury

Knižní zdroje

HOPKINS, Callum. *PHP okamžitě*. Brno: Computer Press, 2014. ISBN 978-80-251-4196-0.

SHARKIE, Craig a Andrew FISHER. *Responzivní webdesign: okamžitě*. Brno: Computer Press, 2015. ISBN 978-80-251-4384-1.

STROUKAL, Dominik a Jan SKALICKÝ. *Bitcoin: peníze budoucnosti : historie a ekonomie kryptoměn, stručná příručka pro úplné začátečníky*. Praha: Ludwig von Mises Institut CZ&SK, 2015. ISBN 978-80-87733-26-4.

Elektronické zdroje

Drahomír Hanák. *IT Network* [online]. [cit. 2018-03-04]. Dostupné z: https://www.itnetwork.cz/nezarazene/stoparuv-pruvodce-rest-api

Jani Tarvainen. *Symfony Finland* [online]. 2015 [cit. 2018-02-26]. Dostupné z: https://www.symfony.fi/entry/symfony-benchmarks-php-56-hhvm-and-php-7

Lukáš Jelínek. *Linux Expres* [online]. 2015 [cit. 2018-03-01]. Dostupné z: https://www.linuxexpres.cz/software/hhvm-co-to-je-proc-pouzivat-jak-na-to

Tadoch. *Trade arena* [online]. 2018 [cit. 2018-03-08]. Dostupné z: https://www.tradearena.cz/rubriky/kryptomeny/jak-funguje-tezba-bitcoinu-podrobne-vysvetleni-miningu_363.html

Taskkill. *IT Network* [online]. [cit. 2018-02-27]. Dostupné z: https://www.itnetwork.cz/javascript/ajax/uvod-do-ajaxu

Tovarna.cz [online]. [cit. 2018-04-22]. Dostupné z: http://www.tovarna.cz/cz/slovnik-pojmu/45-mit/

Seznam zkratek

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

ASIC – Application Specific Integrated Circuit

CMS – Content Management System

CSRF – Cross-site Request Forgery

CSS – Cascading Style Sheets

CSV – Comma-separated values

DMS – Document Management System

FTP – File Transfer Protocol

HHBC – HipHop bytecode

HHVM – HipHop Virtual Machine

HTML – HyperText Markup Language

IDE – Integrated Development Environment

IT – Informační technologie

JSON – JavaScript Object Notation

LESS – Leaner Style Sheets

MIT – Massachusetts Institute of Technology

MySQL – My Structured Query Language

P2P – Peer to Peer

PC – Personal Computer

PDF – Portable Document Format

PHP – Hypertext Preprocessor

QR – Quick Response

REST – Representational State Transfer

SASS – Syntactically Awesome Style Sheets

SPAYD – Short Payment Descriptor

VPS – Virtual Private Server

XML – Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Prohlašuji, že

- jsem byl(a) seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;

- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, bakalářskou práci užít (§ 35 odst. 3);

- souhlasím s tím, že bakalářská práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že bibliografické údaje o bakalářské práci budou zveřejněny v informačním systému VŠB-TUO;

- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;

- bylo sjednáno, že užít své dílo, bakalářskou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne 7.5.2018

.....

Jan Kožušník

Seznam příloh

Příloha 1: Dotazník

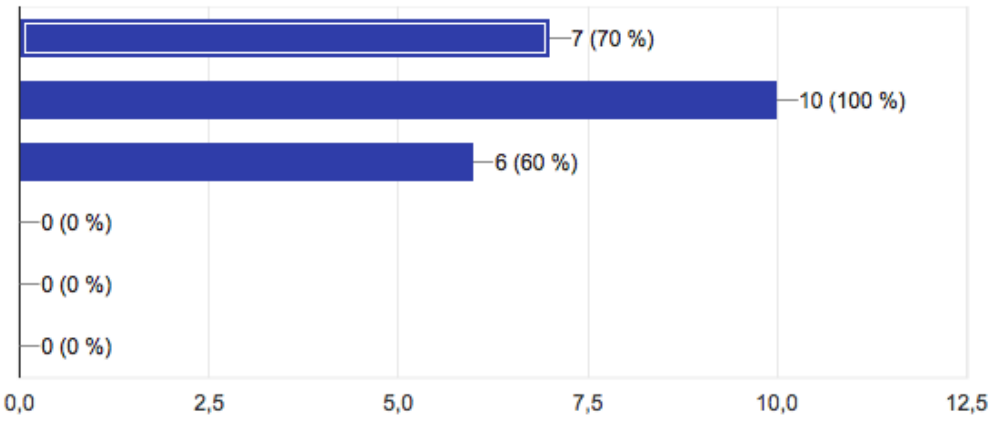
Usnadňuje Vám office Vaši práci?

Obrázek 1 Graf znázorňující odpovědi k dané otázce

Jak byste charakterizovali office?

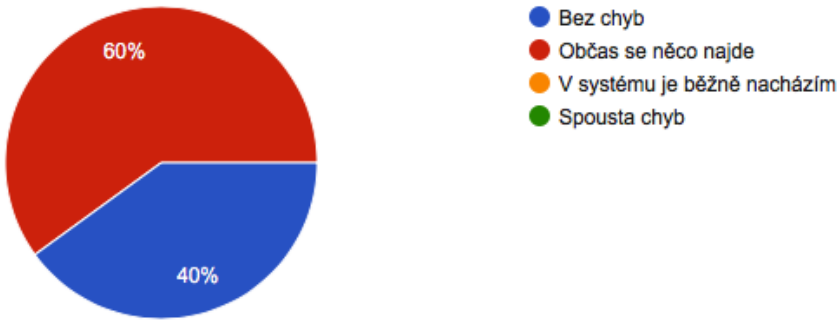
Zde se jednalo o otázku s možností více odpovědí. Možné odpovědi byly v tomto pořadí:

1. Jednoduchý
2. Přehledný
3. Dobře se mi v něm pracuje
4. Zmatený
5. Některé věci jsou zbytečně komplikované
6. Některé věci mi tam chybí



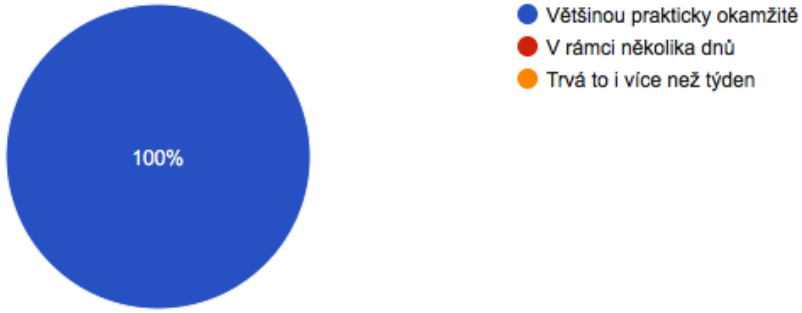
Obrázek 2 Grafické znázornění odpovědí k dané otázce

Jak byste ohodnotili výskyt chyb v office?



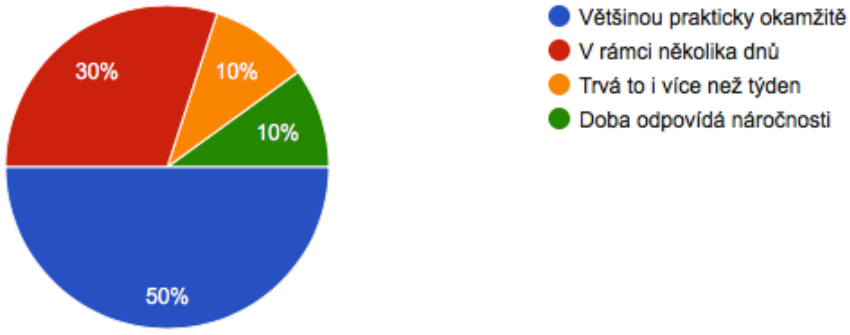
Obrázek 3 Grafické znázornění odpovědí k dané otázce

Jak byste ohodnotili rychlost opravy chyb?



Obrázek 4 Grafické znázornění odpovědí k dané otázce

Jak byste ohodnotili rychlost implementace Vašich požadavků?



Obrázek 5 Grafické znázornění odpovědí k dané otázce

Co se Vám na office líbí?

přehlednost (2)
přehledný
Office je přehledný, takže prakticky hned najdu co hledám. Uživatelsky přívětivé prostředí.
Jednoduchost.
jednoduchý, intuitivní, oceňuji možnost na proklik do mailu
Jednoduchost, přehled
je jednoduše ovladatelný, usnadňuje práci
rychlé a přehledné zpracování a vzhled
Přehlednost, je tam vše, co potřebuji.

Obrázek 6 Zobrazení odpovědí k dané otázce

Co se Vám na office nelíbí?

nic jsem nenašel
Zatím jsem na nic nenarazila.
Nemám žádnou negativní zkušenost.
černá barva
nemám žádné výhrady
moc tmavý
Nic
líbí se mi vše
momentálně mě nic nenapadá
Někdy jsou později připsané pohyby na klientských účtech.

Obrázek 7 Zobrazení odpovědí k dané otázce

Co byste v office změnili, co Vám v něm chybí?

přímý odkaz na zaevidování nového klienta
Zatím jsem na nic nenarazila.
Pár dnů byl k dispozici proklik do mailu,ale nevím proč zmizel.
nevím
zatím nic
-
Přidání více informací, intuitivnější report, upozornění vkladů a upozornění vyberu prostředků.
chtělo by to také měsíční náhled kalendáře pro lepší orientaci v přípravovaných akcích
momentálně nic
Nic mě nenapadá.

Obrázek 8 Zobrazení odpovědí k dané otázce

Příloha 2: CD se zdrojovými kódy

Přiložené CD obsahuje vybrané kódy webové aplikace.